Piecewise training for structured prediction

Mach Learn (2009) 77: 165–194
DOI 10.1007/s10994-009-5112-z

Piecewise training for structured prediction

Charles Sutton · Andrew McCallum

Received: 28 April 2008 / Accepted: 13 April 2009 / Published online: 16 June 2009

Abstract A drawback of structured prediction methods is that parameter estimation requires repeated inference, which is intractable for general structures. In this paper, we present an approximate training algorithm called *piecewise training (PW)* that divides the factors into tractable subgraphs, which we call *pieces*, that are trained independently. Piecewise training can be interpreted as approximating the exact likelihood using belief propagation, and different ways of making this interpretation yield different insights into the method. We also present an extension to piecewise training, called *piecewise pseudolikelihood (PWPL)*, designed for when variables have large cardinality. On several real-world natural language processing tasks, piecewise training performs superior to Besag's pseudolikelihood and sometimes comparably to exact maximum likelihood. In addition, PWPL performs similarly to PW and superior to standard pseudolikelihood, but is five to ten times more computationally efficient than batch maximum likelihood training.

Keywords Graphical models · Conditional random fields · Local training · Belief propagation

1 Introduction

Fundamental to many applications is the ability to predict multiple variables that depend on each other. Such applications are as diverse as classifying regions of an image (Li 2001), estimating the score in a game of Go (Stern et al. 2005), segmenting genes in a strand of

Editors: Charles Parker, Yasemin Altun, and Prasad Tadepalli.

C. Sutton (✉) · A. McCallum
Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA
e-mail: casutton@eecs.berkeley.edu

A. McCallum
e-mail: mccallum@cs.umass.edu

Present address:
C. Sutton
Computer Science Division, University of California, Berkeley, CA 94720, USA

DNA (Bernal et al. 2007), extracting syntax from natural-language text (Taskar et al. 2004b), and determining whether noun phrases in an article are coreferent (McCallum and Wellner 2005). In such applications, we wish to predict a vector $\mathbf{y} = \{y_0, y_1, \ldots, y_T\}$ of random variables given an observed feature vector $\mathbf{x}$. A relatively simple example from natural-language processing is part-of-speech tagging, in which each variable y_s is the part-of-speech tag of the word at position s, and the input $\mathbf{x}$ is divided into feature vectors $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$. Each $\mathbf{x}_s$ contains various information about the word at position s, such as its identity, orthographic features such as prefixes and suffixes, membership in domain-specific lexicons, and information in semantic databases such as WordNet.

An attractive approach to such problems is provided by *structured prediction* methods. Structured prediction methods are essentially a combination of classification and graphical modeling, combining the ability to compactly model multivariate data with the ability to perform prediction using large sets of input features. The idea is, for an input $\mathbf{x}$, to define a discriminant function $\phi(\mathbf{y}, \mathbf{x})$, and predict $\mathbf{y}^* = \arg\max_{\mathbf{y}} \phi(\mathbf{y}, \mathbf{x})$. Just as in graphical modeling, this discriminant function factorizes according to a set of local factors, each of which depend on only a small number of variables. But as in classification, each local factor is modeled as a linear function of $\mathbf{x}$, although perhaps in some induced high-dimensional space. Examples of structured prediction methods include conditional random fields (CRFs) (Lafferty et al. 2001; Sutton and McCallum 2007a), max-margin Markov networks (Taskar et al. 2004a), MIRA (Crammer and Singer 2003), and search-based methods (Daumé and Marcu 2005).

A main drawback to structured prediction methods is that parameter estimation requires repeated inference. This is because parameters are typically chosen by M-estimation, that is, by maximizing the sum of some objective function, such as the log-likelihood or the margin, evaluated at each training instance. Numerical optimization of this objective typically requires performing inference over each data point at many different values in parameter space. For models with tractable structure, such as chains and parse trees, this is often feasible, but for general structures inference is intractable. Therefore, the need for repeated inference during training is a significant challenge to applying structured prediction methods to large-scale data, especially when the model has complex structure. For this reason, approximate training methods for structured models are of great interest.

An attractive family of approximate training methods is *local training* methods, which are training methods that depend on sums of local functions of only a few factors rather than on global functions of the entire graph such as the likelihood. In other words, the objective function can be written as $\ell(\theta; \mathbf{x}, \mathbf{y}) = \sum_R \ell_R(\mathbf{x}_R, \mathbf{y}_R; \theta)$, where R indexes a set of local functions $\{\ell_R\}$, each of which can be computed efficiently from only a few of the factors of the full structured model. The best-known example of a local training method is Besag's pseudolikelihood (Besag 1975), which is a product of the conditional probability of each node given its Markov blanket. The chief attraction of this method is that it achieves consistency without requiring inference in the training procedure. Although in some situations pseudolikelihood can be very effective (Parise and Welling 2005; Toutanova et al. 2003), in other applications, its accuracy can be poor.

In this paper, we present a novel local training method called *piecewise training*, in which the model's factors are divided into possibly overlapping sets of *pieces*, which are each trained separately. Piecewise training is based on the intuition that if all of the local factors fit the data well, then the resulting global distribution is likely to be reasonable. At test time, the resulting weights are used just as if they had been trained using maximum likelihood, that is, on the unseen data they are used to predict the labels using a standard approximate inference algorithm, such as max-product belief propagation. This method has been employed

occasionally throughout the literature, but to our knowledge its properties have never been systematically examined. When using piecewise training, the modeler must decide how to split the model into pieces before training. In this paper, we focus on the factor-as-piece approximation, in which each factor of the model is placed in a separate piece.

We motivate this procedure as an approximation to the likelihood of a conditional random field. In particular, this training procedure can be viewed in several ways. First, separate training of each piece can be accomplished by numerically maximizing an approximation to the likelihood. This approximate likelihood can be seen as the true likelihood on a transformation of the original graph, which we call the node-split graph, in which each of the pieces is an isolated component. Second, another view is based on belief propagation (BP); namely, the objective function of piecewise training is the same as the BP approximate likelihood with uniform messages, as if BP has been stopped after zero iterations. Finally, the third view is based on a set of estimating equations that generalize the pseudo-moment matching estimator of Wainwright et al. (2003b). These three viewpoints are useful both for understanding these algorithms, and for designing extensions of these methods.

Additionally, we consider an extension to piecewise training, tailored for the case in which the variables have large cardinality. When variables have large cardinality, training can be computationally demanding even when the model structure is tractable. For example, consider a series of processing steps of a natural-language sentence (Sutton et al. 2004; Finkel et al. 2006), which might begin with part-of-speech tagging, continue with more detailed syntactic processing, and finish with some kind of semantic analysis, such as relation extraction or semantic entailment. This series of steps might be modeled as a simple linear chain, but each variable has an enormous number of outcomes, such as the number of parses of a sentence. In such cases, even training using forward-backward is infeasible, because it is quadratic in the variable cardinality. Thus, we also desire approximate training algorithms not only that are subexponential in the model's treewidth, but also that scale well in the variable cardinality. The Besag pseudolikelihood (PL) is attractive here, because its running time is linear in the variable cardinality. However, piecewise training performs significantly better than pseudolikelihood on the real-world data considered here, but unlike pseudolikelihood it does not scale well in the variable cardinality.

To address this problem, we introduce and analyze a hybrid method, called *piecewise pseudolikelihood* (PWPL), that combines the advantages of both approaches. Essentially, while pseudolikelihood conditions each variable on all of its neighbors, PWPL conditions only on those neighbors within the same piece of the model, for example, that share the same factor. This is illustrated in Fig. 2. Thus, PWPL can be viewed as double approximation: Rather than performing maximum likelihood on the node-split graph, as regular PW does, PWPL performs pseudolikelihood on the node-split graph. Remarkably, this double approximation performs better than the pseudolikelihood approximation alone on several real-world natural-language processing (NLP) data sets. In other words, PWPL behaves more like PW than like pseudolikelihood, in terms of the prediction accuracy of the estimated model. The training speed-up of PWPL can be significant even in linear-chain models, because forward-backward training is quadratic in the variable cardinality.

In the remainder of this paper, we define piecewise training (Sect. 3.1), explaining it from the perspectives of the node-split graph (Sect. 3.2) and of belief propagation (Sects. 3.3 and 3.4). Also, we show that in some cases, the piecewise likelihood is a simple lower bound on the true likelihood (Sect. 3.5). Then we present PWPL (Sect. 4.1), describing it in terms of the node-split graph. This viewpoint allows us to show that under certain conditions, PWPL converges to the piecewise solution in the asymptotic limit of infinite data (Sect. 4.2). In addition, it provides some insight into when PWPL may be expected to

do well and to do poorly, an insight that we verify on synthetic data (Sect. 5.2.1). Finally, we apply both PW and PWPL to several natural-language data sets. First, piecewise training has better accuracy than pseudolikelihood and is sometimes comparable to exact maximum likelihood (Sect. 5.1). Second, PWPL performs often comparably to piecewise training and to maximum likelihood, and has higher accuracy than pseudolikelihood on all of our data sets (Sect. 5.2.2). Furthermore, PWPL can be as much as ten times faster than batch CRF training.

2 Background

This section presents background on structured prediction and pseudolikelihood. In this paper, we are interested in estimating the conditional distribution $p(\mathbf{y}|\mathbf{x})$ of a discrete output vector $\mathbf{y}$ given an input vector $\mathbf{x}$. We model p by a factor graph G with variables $s \in Y$ and factors $\{\Psi_a\}$, where a is a member of some index set $\mathcal{F}$. Each factor has a domain of some subset of variables $Y_a \subset Y$ from the output and X_a from the input. We write the value of Ψ_a on some assignment $(\mathbf{y}, \mathbf{x})$ as $\Psi_a(\mathbf{y}_a, \mathbf{x}_a)$. With an abuse of notation, we will also use $s \in a$ as a shorthand for $s \in Y_a$.

We model the conditional distribution as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\theta; \mathbf{x})} \prod_{a \in \mathcal{F}} \Psi_a(\mathbf{y}_a, \mathbf{x}_a), \tag{1}$$

where $Z(\theta; \mathbf{x})$ is a normalization constant, which is also called the *partition function*. A conditional distribution which factorizes in this way is called a *conditional random field (CRF)* (Lafferty et al. 2001; Sutton and McCallum 2007a). Typically, each factor is modeled in an exponential form

$$\Psi_a(\mathbf{y}_a, \mathbf{x}_a) = \exp\{\theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a)\}, \tag{2}$$

where θ_a is real-valued parameter vector, and f_a returns a vector of *features* or sufficient statistics over the variables in the set a. The parameters of the model are the set $\theta = \{\theta_a\}_{a \in \mathcal{F}}$, and we will be interested in estimating them given a sample of fully observed input-output pairs $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$. Throughout, we will also talk about the empirical distributions

$$\tilde{p}(\mathbf{x}) = N^{-1} \sum_{i=1}^{N} \mathbf{1}_{\{\mathbf{x}=\mathbf{x}^{(i)}\}},$$

$$\tilde{p}(\mathbf{y}|\mathbf{x}) = N^{-1}[\tilde{p}(\mathbf{x})]^{-1} \sum_{i=1}^{N} \mathbf{1}_{\{\mathbf{y}=\mathbf{y}^{(i)}, \mathbf{x}=\mathbf{x}^{(i)}\}}.$$

A commonly used estimator of θ is maximum likelihood. The log-likelihood is $\ell(\theta) = \sum_{i=1}^N \ell(\theta; \mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, where

$$\ell(\theta; \mathbf{x}, \mathbf{y}) = \sum_{a \in \mathcal{F}} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) - A(\theta; \mathbf{x}), \tag{3}$$

$$A(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}} \exp\left\{ \sum_{a \in \mathcal{F}} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) \right\}. \tag{4}$$

Maximum likelihood estimation is intractable for general graphs, so for such graphs parameter estimation is performed approximately. One approach is to approximate the log partition

function $A(\theta; \mathbf{x})$ directly, such as by Monte Carlo or variational methods. In this paper, we take a different approach, which is to maximize an approximation of the likelihood that is more computationally tractable.

An example of a computationally convenient objective for estimation is the pseudolikelihood of Besag (1975). Pseudolikelihood simultaneously classifies each node given its neighbors in the graph. For a variable s, let $N(s)$ be the set of all of its neighbors, not including s itself. Then the log-pseudolikelihood is defined as

$$\ell_{\text{PL}}(\theta; \mathbf{x}, \mathbf{y}) = \sum_{s \in Y} \log p(y_s | \mathbf{y}_{N(s)}, \mathbf{x}),$$

where the conditional distributions are

$$p(y_s | \mathbf{y}_{N(s)}, \mathbf{x}) = \frac{\prod_{a \ni s} \Psi_a(y_s, \mathbf{y}_{N(s)}, \mathbf{x}_a)}{\sum_{y'_s} \prod_{a \ni s} \Psi_a(y'_s, \mathbf{y}_{N(s)}, \mathbf{x}_a)}, \tag{5}$$

where $a \ni s$ means that the product is to range over of set of all indices a of factors that depend on the variable s. In other words, this is a sum of conditional log likelihoods, where for each variable we condition on the true values of its neighbors in the training data.

It is a well known result that if the model family includes the true distribution, then pseudolikelihood converges to the true parameter setting in the limit of infinite data (Gidas 1988; Hyvarinen 2006). Intuitively, one way to understand this is that pseudolikelihood attempts to match all of the model conditional distributions to those of the empirical distribution. If it succeeds in matching them all exactly, then in the limit of infinite data, the empirical conditional distributions will equal the true conditional distributions, so the Gibbs sampler of the model will be the same as the Gibbs sampler of the true distribution.

3 Piecewise training

In this section, we present piecewise training. The motivation is that in some applications, the local information in each factor alone, without performing inference, is enough to do fairly well at predicting the outputs, but some amount of global information can still help. Therefore, to reduce training time, it makes sense to perform less inference at training time than at test time, because at training time we loop through the examples repeatedly, whereas at test time we only need to make each prediction once. For example, suppose we want to train a loopy pairwise Markov random field (MRF). In piecewise estimation, we train the parameters of each edge independently, as if each edge were a separate two-node MRF of its own. Finally, on test data, the parameters resulting from this local training become the parameters used to perform global inference, using some standard approximate inference algorithm.

In the rest of this section, we first define piecewise training more formally (Sect. 3.1). Then we describe piecewise training from four different viewpoints. Different viewpoints provide different insights into when piecewise training can be expected to perform well, and suggest different ways to generalize the method. Piecewise training can be viewed

1. as maximizing the likelihood on a transformation of the original graph (Sect. 3.2),
2. as performing a type of *pseudo-moment matching*, generalizing a previous estimator due to Wainwright et al. (2003b) (Sect. 3.3),
3. as maximizing an approximation to the likelihood resulting from a dual version of the Bethe free energy (Sect. 3.4),
4. as maximizing a lower bound on the true likelihood (Sect. 3.5).

3.1 Definition

Now we define the piecewise estimator more generally. Assume that the model's factors are divided into a set $\mathcal{P} = \{R_0, R_1, \ldots\}$ of pieces; each piece $R \in \mathcal{P}$ is a set of factor indices $R \subset \mathcal{F}$. The pieces need not be disjoint. For example, in a grid-shaped MRF with unary and pairwise factors, we might isolate each factor in its own piece, or alternatively we might choose one piece for each row and each column of the MRF, in which case each unary factor would be shared between its row piece and its column piece.

To train the pieces separately, each piece R has a local log-likelihood

$$\ell_R(\theta) = \sum_{a \in R} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) - A_R(\theta; \mathbf{x}), \tag{6}$$

where $A_R(\theta; \mathbf{x})$ is the *local log partition function* for the piece, that is,

$$A_R(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}_R} \exp\left\{ \sum_{a \in R} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) \right\}, \tag{7}$$

where $\mathbf{y}_R$ is the vector of variables used anywhere in piece R. This is the log-likelihood for the piece R if it were a completely separate graphical model. If the pieces are disjoint, and no parameters are shared between distinct factors, then we could train each piece by separately computing parameters $\theta^R_{\text{PW}} = \max_{\theta_R} \ell_R(\theta_R)$. But in order to handle parameter tying and overlapping pieces, we instead perform a single optimization, maximizing the sum of all of the single-piece log-likelihoods. So for a set $\mathcal{P}$ of pieces, the piecewise log-likelihood becomes

$$\ell_{\text{PW}}(\theta; \mathbf{x}, \mathbf{y}) = \sum_{R \in \mathcal{P}} \sum_{a \in R} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) - \sum_{R \in \mathcal{P}} A_R(\theta; \mathbf{x}). \tag{8}$$

For example, consider the special case of per-edge pieces in a pairwise MRF with no tied parameters. Then, for an edge (s, t), we have $A_{st}(\theta) = \log \sum_{y_s, y_t} \Psi(y_s, y_t)$, so that piecewise training corresponds exactly to training independent probabilistic classifiers on each edge.

Now let us compare the piecewise likelihood to the exact likelihood (3). Notice that the only difference between the piecewise log-likelihood ℓ_{PW} and the exact log-likelihood ℓ is in the second summation of (8). So $A_{\text{PW}}(\theta; \mathbf{x}) := \sum_R A_R(\theta; \mathbf{x})$ can be viewed as an approximation of the log partition function $A(\theta; \mathbf{x})$. By using piecewise training, we need to compute only local normalization over small cliques, which for loopy graphs is potentially much more efficient.

A choice of pieces to which we devote particular attention is the factor-as-piece approximation, in which each factor in the model is assigned to its own piece, that is, $\mathcal{P} = \{\{a\} \mid a \in \mathcal{F}\}$. Unless specified otherwise, for the rest of this paper we will assume the factor-as-piece approximation.

There is a potential ambiguity in this choice. To see this, write out the summation contained within the dot product of (2)

$$\Psi_a(\mathbf{y}_a, \mathbf{x}_a) = \exp\left\{ \sum_k \theta_{ak} f_{ak}(\mathbf{y}_a, \mathbf{x}_a) \right\},$$

so that each factor has multiple parameters and sufficient statistics. But we could just as well place each sufficient statistic in a factor all to itself, that is,

$$\Psi_{ak}(\mathbf{y}_a) = \exp\{\theta_{ak} f_{ak}(\mathbf{y}_a)\}, \tag{9}$$

and define the pieces at that level of granularity. Such a fine-grained choice of pieces could be useful. For example, in a linear-chain model, we might choose to view the model as a weighted finite-state machine, and partition the state-transition diagram into pieces. In this paper, however, when we use the factor-as-piece approximation, we will not use the fine-grained factorization of (9), that is, we will assume that the graph has been constructed so that no two factors share exactly the same support.

3.2 The node-split graph

The piecewise log-likelihood (8) can be viewed as the exact log-likelihood in a transformation of the original graph. In the transformed graph, we split the variables, adding one copy of each variable for each factor that it participates in, as pictured in Fig. 1. We call the transformed graph the *node-split graph*.

Formally, the splitting transformation is as follows. Given a factor graph G, create a new graph G' with variables $\{V_{as}\}$, where $a \in \mathcal{F}$ and $s \in a$. For any factor Ψ_a, let π_a map variables in G to their copy in G', that is, $\pi_a(s) = V_{as}$ for any variable s in G. Finally, for each factor $\Psi_a(y_a, \theta)$ in G, add a factor Ψ'_a to G' as

$$\Psi'_a(\pi_a(y_a), \theta) = \Psi_a(y_a, \theta). \tag{10}$$

If we wish to use pieces that are larger than a single factor, then the definition of the node-split graph can be modified accordingly.

Clearly, piecewise training in the original graph is equivalent to exact maximum likelihood training in the node-split graph. This view of piecewise training will prove useful for understanding PWPL in Sect. 4.

This viewpoint also makes clear the bias in the piecewise estimate. Suppose that the model family contains the true distribution, and let $p^*(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{x}; \theta^*)$ be the true distribution of the data. Also, let $p(\mathbf{y}_a|\mathbf{x}; \theta^*)$ be the marginal distribution of the variables in factor a according to the true distribution.

The piecewise likelihood cannot distinguish p^* from the distribution p_{NS} over the original space that is defined by the product of marginals

$$p_{\text{NS}}(\mathbf{y}|\mathbf{x}) \propto \prod_{a \in G'} p(\mathbf{y}_a|\mathbf{x}; \theta^*). \tag{11}$$

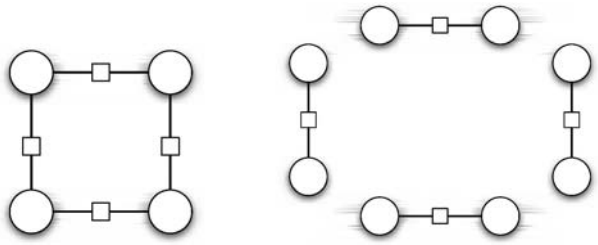

Fig. 1 Example of node splitting. *Left* is the original model, *right* is the version trained by piecewise. In this example, there are no unary factors

By that we mean that if the empirical distribution $\tilde{p}$ were exactly p^*, as in the limit of infinite data, then the piecewise likelihood $\ell_{\text{PW}}(\theta)$ for any θ would be exactly the same as if $\tilde{p} = p_{\text{NS}}$.

Then we can express the bias in the piecewise estimate as follows. Suppose there exists a parameter setting θ_{NS} of the model such that $p_{\text{NS}}(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{x}; \theta_{\text{NS}})$ for all $\mathbf{y}, \mathbf{x}$. Then the piecewise estimate converges to θ_{NS} (rather than θ^*) in the infinite data limit. For example, suppose that, unknown to the modeler, all the variables in the graph are actually independent. Then $p_{\text{NS}}(\mathbf{y}|\mathbf{x}) = \prod_{s \in Y} p(y_s|\mathbf{x})^{d(s)}$, where $d(s)$ is the degree of variable s, and the piecewise estimate will converge to this distribution in the limit of infinite data rather than the true distribution. Essentially, PW overcounts the effect of the variables that share multiple factors.

3.3 Pseudo-moment matching viewpoint

Piecewise training is based on the intuition that if all of the local factors fit the data well, then the resulting global distribution is likely to be reasonable. An interesting way of formalizing this idea is by way of the *pseudo-moment matching* estimator of Wainwright et al. (2003b). In this section, we show that there is a sense in which piecewise training can be viewed as an extension of the pseudo-moment matching estimator.

First, consider the case in which a discrete distribution $p(\mathbf{y})$ factorizes according to a graph with fully-parameterized tables, that is,

$$\Psi_a(\mathbf{y}_a) = \exp\left\{\sum_{\mathbf{y}'_a} \theta(\mathbf{y}'_a)\mathbf{1}_{\{\mathbf{y}_a = \mathbf{y}'_a\}}\right\}. \tag{12}$$

Here we are not (yet) conditioning on any input variables. Let $\tilde{p}(\mathbf{y})$ be the empirical distribution, that is, $\tilde{p}(\mathbf{y}) \propto \sum_i \mathbf{1}_{\{\mathbf{y}=\mathbf{y}^{(i)}\}}$.

The pseudo-moment matching estimator chooses parameters that maximize the BP likelihood, remarkably, without actually computing any of the message updates. This estimator is

$$\begin{aligned} \hat{\theta}_a(\mathbf{y}_a) &= \log \frac{\tilde{p}(\mathbf{y}_a)}{\prod_{s \in a} \tilde{p}(y_s)}, \\ \hat{\theta}_s(y_s) &= \log \tilde{p}(y_s). \end{aligned} \tag{13}$$

For these parameters, there exists a set of messages that (a) are a fixed point of BP, and (b) the resulting beliefs q_a and q_s equal the empirical marginals.

This can be shown using a different viewpoint on BP, namely the reparameterization perspective due to Wainwright et al. (2003a). In this view, the BP updates are expressed solely in terms of the beliefs b^n at each iteration n of the algorithm. The beliefs are initialized as $b^0_a \propto \Psi_a$ for all factors, and $b^0_s \propto 1$ for all variables. The updates at iteration n are

$$\begin{aligned} b^n_s(y_s) &= \sum_{\mathbf{y}_{N(s)}} B^{n-1}_s(y_s, \mathbf{y}_{N(s)}), \\ b^n_a(\mathbf{y}_a) &= \sum_{\mathbf{y}_{N(a)}} B^{n-1}_a(\mathbf{y}_a, \mathbf{y}_{N(a)}) \end{aligned} \tag{14}$$

where the distributions B_a^{n-1} and B_s^{n-1} are defined as

$$\begin{aligned} B_s^{n-1}(y_s, \mathbf{y}_{N(s)}) &\propto b^{n-1}(y_s) \prod_{a \ni s} \frac{b_a^{n-1}(\mathbf{y}_a)}{b_s^{n-1}(y_s)}, \\ B_a^{n-1}(\mathbf{y}_a, \mathbf{y}_{N(a)}) &\propto b_a^{n-1}(\mathbf{y}_a) \prod_{s \in a} \prod_{c \ni s \setminus a} \frac{b_c^{n-1}(\mathbf{y}_c)}{b_s^{n-1}(y_s)}. \end{aligned} \tag{15}$$

(This notation is adapted from Rosen-Zvi et al. (2005).) Here $N(a) = \bigcup_{s \in a} N(s) - Y_a$. It can be shown that the beliefs from the message-based recursions (that is, the standard form of BP) are equal to those from the reparameterization-based recursions (14). To see this, substitute the definition of the beliefs, $b_s^n = \prod_{a \ni s} m_{as}^n$ and $b_a^n = \Psi_a \prod_{s \in a} m_{sa}$ into (14), and cancel factors.

One consequence of this perspective on BP is that it is possible to see immediately that the pseudo-moment matching equations (13) are correct. This is because under that parameter setting, once we set $b_a^0 \propto \Psi_a$, those beliefs are already at a fixed point, and also are equal to the empirical marginals.

This discussion of pseudo-moment matching, however, focused solely on a joint model $p(\mathbf{y})$, in which we had no input $\mathbf{x}$. For conditional random fields, on the other hand, we are interested in estimating the parameters of conditional distributions $p(\mathbf{y}|\mathbf{x})$. Furthermore, in virtually all practical applications the parameters of different factors are tied, analogous to the way that the same transition distribution is often used for each time step in an HMM. So as originally presented, the pseudo-moment matching estimator is not a practical learning algorithm for CRFs (or, in most cases for generative MRFs, which often employ parameter tying).

A simple generalization is to require for all inputs $\mathbf{x}$ in the training set that

$$\begin{aligned} \Psi_a(\mathbf{y}_a, \mathbf{x}) &= \frac{\tilde{p}(y_a|\mathbf{x})}{\prod_{s \in a} \tilde{p}(y_s|\mathbf{x})}, \\ \Psi_s(y_s, \mathbf{x}) &= \tilde{p}(y_s|\mathbf{x}). \end{aligned} \tag{16}$$

However, this does not handle the fact that parameters are tied across different factors in $p(\mathbf{y}|\mathbf{x})$ for a fixed input $\mathbf{x}$, or that the same parameters are used for $p(\mathbf{y}|\mathbf{x})$ across different inputs. In other words, the factor values of $\Psi_a(\cdot, \mathbf{x})$ do not have an independent degree of freedom for each input value $\mathbf{x}$.

A natural approach to further generalize the pseudo-moment matching idea (which is, as far as we know, novel) is to treat (16) as a nonlinear set of equations to be solved. To do this, we optimize the objective function

$$\min_\theta \sum_{a \in \mathcal{F}} D\left(\Psi_a \| \tilde{p}_a \prod_{s \in a} \tilde{p}_s^{-1}\right) + \sum_{s \in Y} D(\Psi_s \| \tilde{p}_s), \tag{17}$$

where $D(\cdot\|\cdot)$ is a divergence measure. By a *divergence measure* $D(p\|q)$, we simply mean a nonnegative function that is 0 if and only if $p = q$. Then if a parameter setting θ exists such that the divergence is zero, then the equations have been solved exactly, and the parameters θ optimize the BP approximation to the likelihood.

Furthermore, in the setting of structured prediction, it is usually the case that each distinct input in the training data occurs only once. For example, this is the case in all the natural-language data sets considered in Sect. 5: Since there are exponentially many possible English

sentences, it is highly unlikely to see the same sentence more than once. In this case, for every pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ in the training data, the empirical distribution $\tilde{p}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) = 1$, which implies that $\tilde{p}_a \prod_{s \in a} \tilde{p}_s^{-1} = \tilde{p}_a$. Then the objective (17) is equal to

$$\min_{\theta} \sum_{a \in \mathcal{F}} D(\Psi_a \| \tilde{p}_a) + \sum_{s \in Y} D(\Psi_s \| \tilde{p}_s). \tag{18}$$

This provides another view of piecewise training. Suppose we select as the divergence

$$D(\Psi_a \| \tilde{p}_a) = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \left[\sum_{\mathbf{y}_a} \tilde{p}(\mathbf{y}_a|\mathbf{x}) \log \frac{\tilde{p}(\mathbf{y}_a|\mathbf{x})}{\Psi_a(\mathbf{y}_a, \mathbf{x}_a)} + \log \sum_{\mathbf{y}_a} \Psi_a(\mathbf{y}_a, \mathbf{x}_a) \right]. \tag{19}$$

This is the Kullback-Leibler divergence between $\tilde{p}_a$ and Ψ_a, when Ψ_a is normalized as if it were a probability distribution over $\mathbf{y}_a$. For this choice of D, minimizing the divergence in (18) yields an equivalent optimization problem to the piecewise log-likelihood (8). Furthermore, if the optimal value of the divergence is 0, then the corresponding parameters are a solution to (16), and so the factors satisfy the pseudo-moment matching condition separately for each $\mathbf{x}$ in the training data. This provides a justification of the intuition that fitting locally can lead to a reasonable global solution: it is not the case that fitting factors locally causes the true marginals to be matched to the empirical distribution, but it does cause the BP approximation to the marginals to be matched to the empirical distribution, over the inputs that were observed in the training data.

If the assumption in the previous paragraph does not hold—namely, if some inputs appear multiple times in the training data with different outputs—then it is no longer the case that $\tilde{p}_a \prod_{s \in a} \tilde{p}_s^{-1} = \tilde{p}_a$, so the pseudo-moment equations (17) are no longer equivalent to the piecewise likelihood, but are closely analogous. Furthermore, in this case the value and gradient of (17) can be still be computed locally, without the need for computing any marginal distributions, so it can optimized using gradient-based methods in exactly the same way as the piecewise likelihood. It is possible that (17) may be preferable to the piecewise likelihood in this situation, but this situation seems sufficiently uncommon that we do not consider it further.

3.4 Viewpoint from the dual Bethe energy

Another way of understanding piecewise training arises from belief propagation. The main idea is that the piecewise likelihood can be derived from the true likelihood by approximating the true partition function by the value of the Bethe free energy after zero iterations of BP, that is, when all of the messages are uniform. In the rest of this section, we first explain the Bethe dual energy, and then explain how it can be used to derive the piecewise likelihood.

A variational inference method is one that converts an inference problem into an optimization problem. Although loopy BP was originally described algorithmically, in a way that does not appear to be optimizing any objective function, in fact BP can be viewed as a variational inference method. This is because any BP fixed point is also stationary points of a function called the *Bethe free energy* (Yedidia et al. 2005). For our purposes, a *free energy* can be thought of simply as an objective function over tractable probability distributions q, that measures how well they approximate an intractable distribution p. Other free energies, distinct from the Bethe energy, can also have the property of matching the fixed points of BP. One such free energy, which we will call the *dual Bethe energy* (Minka 2001a, 2005), is particularly useful for understanding piecewise training.

This dual energy arises from the expectation propagation view of BP. Suppose that we approximate each factor Ψ_a by a product of functions m_{as} that each depend on only one variable s. Call this approximate factor $\tilde{\Psi}_a$, so that

$$\Psi_a(\mathbf{y}_a) \approx \tilde{\Psi}_a(\mathbf{y}_a) = \prod_{s \in a} m_{as}(y_s). \tag{20}$$

Approximating each factor yields an approximation to the full distribution p, namely, $p(\mathbf{y}) \approx q(\mathbf{y}) := \prod_a \tilde{\Psi}_a(\mathbf{y}_a)$. Observe that q is possibly unnormalized. (In this section, we suppress the dependence of p and Ψ_a on the input $\mathbf{x}$, in order to simplify the notation.)

Belief propagation can be viewed as incrementally refining this factorized approximation. In this viewpoint, each of the factors m_{as} that make up q is viewed as the BP message from Ψ_a to s. In other words, we view the outgoing messages from each factor as approximating it. Then each message-passing update of loopy BP can viewed as refining one of the terms $\tilde{\Psi}_a$ so that q is closer to p in terms of KL divergence.

This viewpoint can be used to derive a free energy for BP. Since q was chosen to approximate p, it makes sense to use the normalizing constant of q to approximate the normalizing constant of p. More precisely, let p' be the unnormalized version of p, that is, $p'(\mathbf{y}) = \prod_a \Psi_a(\mathbf{y}_a)$. Then define rescaled versions of $\tilde{\Psi}_a$ and q as

$$\bar{\Psi}_a(\mathbf{y}_a) = z_a \tilde{\Psi}_a(\mathbf{y}_a), \tag{21}$$

$$\bar{q}(\mathbf{y}) = \prod_a \bar{\Psi}_a(\mathbf{y}_a), \tag{22}$$

where z_a is a real-valued scaling factor that will be chosen as part of the approximation. The idea is to choose these scaling factors so that the normalizing constant of $\bar{q}$, that is, $\sum_{\mathbf{y}} \bar{q}(\mathbf{y})$, matches as closely as possible the normalizing constant of p', that is, $\sum_{\mathbf{y}} p'(\mathbf{y})$. This can be done by optimizing local divergences in an analogous manner to expectation propagation (EP). Define $\bar{q}^{\setminus a}$ as the distribution $\bar{q}$ but with the factor $\tilde{\Psi}_a$ removed, that is

$$\bar{q}^{\setminus a} = \prod_{b \in \mathcal{F} \setminus a} \bar{\Psi}_b(\mathbf{y}_b). \tag{23}$$

Then one criterion for choosing optimal scaling factors z_a is

$$\min_{z_a} \mathrm{KL}(\Psi_a(\mathbf{y}_a)\bar{q}^{\setminus a}(\mathbf{y}_a) \| z_a \tilde{\Psi}_a(\mathbf{y}_a)\bar{q}^{\setminus a}(\mathbf{y}_a)). \tag{24}$$

Observe that because $\bar{q}^{\setminus a}$ depends on all of the scale factors z_b for all factors b, the local objective function depends on all of the other scale factors as well. The optimal z_a is given by

$$z_a = \frac{\sum_{\mathbf{y}} \frac{\Psi_a(\mathbf{y}_a)}{\tilde{\Psi}(\mathbf{y}_a)} q(\mathbf{y})}{\sum_{\mathbf{y}} q(\mathbf{y})}. \tag{25}$$

Thus the optimal z_a actually does not depend on the other scale values. Now taking the integral $\sum_{\mathbf{y}} \bar{q}(\mathbf{y})$ yields the following approximation to the partition function

$$A_{\textsc{BetheDual}} = \log \prod_{s \in Y} \left(\sum_{y_s} q_s(y_s) \right)^{1-d(s)} \prod_{a \in \mathcal{F}} \left(\sum_{\mathbf{y}_a} \frac{\Psi_a(\mathbf{y}_a)}{\tilde{\Psi}(\mathbf{y}_a)} q(\mathbf{y}_a) \right). \tag{26}$$

It can be shown (Minka 2001b) that this is also a free energy for BP, that is, that fixed points of BP are stationary points of this objective.

Now that we have explained the dual Bethe energy, we show how it can be used to derive the piecewise likelihood. Consider the uniform message setting, that is, $m_{as} = 1$ for all $a \in \mathcal{F}$ and $s \in Y$. This is a common initialization for BP. For this message setting the unnormalized beliefs are $q(\mathbf{y}) = 1$ for all $\mathbf{y}$, and the approximate partition function is

$$A_{\mathrm{BP}}(\theta) = \prod_{a \in \mathcal{F}} \left(C_a \sum_{\mathbf{y}_a} \Psi_a(\mathbf{y}_a, \theta) \right) \prod_{s \in Y} C_s^{1-d(s)}, \tag{27}$$

where C_a and C_s are constants that do not depend on θ. This approximate partition function is the same as that used by piecewise training (under the factor-as-piece approximation), up to a multiplicative constant that does not change the optima. So another view is that piecewise training approximates the likelihood using belief propagation, except that we cut off BP after 0 iterations.

Sutton and Minka (2006) refer to this as a *pseudomarginal viewpoint* on a local training algorithm. This terminology refers to a training method that approximates $A(\theta; \mathbf{x})$ by some function $\tilde{A}$, and chooses parameters to maximize the resulting approximate likelihood. To understand the term "pseudomarginal", consider the derivatives $\partial A / \partial \theta_a$ of the true partition function. If the model contains unary factors of the form $\Psi_s(y_s) = \exp\{\sum_{y_s'} \theta_{s,y_s'} \mathbf{1}_{\{y_s = y_s'\}}\}$ (for scalar $\theta_{s,y_s'}$), then the corresponding partial derivatives of the true partition function yield marginal distributions. Once the partition function is approximated by some $\tilde{A}$, maximizing the approximate likelihood has the effect of matching the pseudomarginals to the empirical marginals.

An alternative to the pseudomarginal viewpoint is a *belief viewpoint* on an approximate training algorithm. Here the idea is to directly approximate the likelihood gradient. The partial derivatives of the log-likelihood are

$$\frac{\partial \ell}{\partial \theta_a} = \sum_{i=1}^{N} f_a(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}) - \sum_{i=1}^{N} \sum_{\mathbf{y}_a} p(\mathbf{y}_a | \mathbf{x}^{(i)}) f_a(\mathbf{y}^{(i)}, \mathbf{x}^{(i)}). \tag{28}$$

(To simplify notation, here we assume that θ_a and f_a are scalar-valued.) Now any approximate inference algorithm can be used to estimate θ, by substituting the approximate marginals returned by the algorithm in place of the exact marginals $p(\mathbf{y}_a | \mathbf{x}^{(i)})$ in (28).

The pseudomarginal and belief viewpoints are distinct: Although every approximate likelihood yields approximate gradients through the pseudomarginals, not all approximate gradients can themselves be obtained as the exact gradient of an approximate objective function. Recently, training methods that have objective function have received much attention, but it is not clear that they should necessarily be preferred. Even for the Bethe energy, the pseudomarginals are distinct from the beliefs. Unlike the beliefs, the pseudomarginals incorporate the derivatives of the messages with respect to the model parameters because of the chain rule. (Another potentially confusing point is that for the primal Bethe energy, the pseudomarginals always equal the beliefs, but this is not true of the dual energy.)

Since the piecewise likelihood arises from the dual Bethe energy at 0 BP iterations, it is natural to ask whether using more iterations would work better. This idea is pursued by Sutton and Minka (2006). In this case, the distinction between beliefs and pseudomarginals becomes important.

3.5 Approximation to the likelihood

A final rationale for the piecewise estimator is that it bounds the likelihood.

Proposition 1 *If the set of pieces $\mathcal{P}$ is a partition of $\mathcal{F}$, then the piecewise partition function is an upper bound on the true partition function*:

$$A(\theta; \mathbf{x}) \leq \sum_{R \in \mathcal{P}} A_R(\theta; \mathbf{x}). \tag{29}$$

Proof The bound is immediate upon expansion of $A(\theta; \mathbf{x})$.

$$A(\theta; \mathbf{x}) = \log \sum_{\mathbf{y}} \prod_{R \in \mathcal{P}} \exp\left\{ \sum_{a \in R} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) \right\} \tag{30}$$

$$\leq \log \prod_{R \in \mathcal{P}} \sum_{\mathbf{y}_R} \exp\left\{ \sum_{a \in R} \theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a) \right\} \tag{31}$$

$$= \sum_{R \in \mathcal{P}} A_R(\theta; \mathbf{x}). \tag{32}$$

The first step (30) follows from the definition of A and the fact that the pieces are disjoint. The bound from (30) to (31) is justified by considering the expansion of the product in (31). The expansion contains every term of the summation in (30), and all terms are nonnegative. □

Therefore, the piecewise likelihood is a lower bound on the true likelihood. If the graph is connected, however, then the bound is not tight, and in practice it is extremely loose.

This bound is actually a special case of the tree-reweighted bounds of Wainwright et al. (2002), a connection which suggests generalizations of the simple piecewise training procedure. In that work, an upper bound on $A(\theta; \mathbf{x})$ was obtained using Jensen's inequality by writing the original parameters θ as a mixture of parameter vectors $\theta = \sum_b \mu_b \theta_b$, where each of the θ_b is chosen to correspond to a tractable subgraph in which inference can be performed efficiently, such as a spanning tree.

For the purpose of extending piecewise training, we consider here the case in which each tractable subgraph consists of a single piece and its associated variables. Let μ be a strictly positive probability distribution over pieces. To use Jensen's inequality, rewrite θ as

$$\theta = \sum_{R \in \mathcal{P}} \mu_R \frac{\theta|_R}{\mu_R}, \tag{33}$$

where $\theta|_R$ is the restriction of θ to R; that is, $\theta|_R$ has the same entries and dimensionality as θ, but with zeros in all entries that are not included in the piece R. Therefore, we immediately have the bound

$$A(\theta) \leq \sum_{R \in \mathcal{P}} \mu_R A\left(\frac{\theta|_R}{\mu_R} \right). \tag{34}$$

This *reweighted piecewise* bound is clearly related to the basic piecewise bound in (29), because $A(\theta|_R)$ differs from $A_R(\theta)$ only by an additive constant which is independent of θ.

Note that the reweighted bound still holds if the pieces are not disjoint, unlike the bound in Proposition 1.

This connection suggests a generalization of the basic piecewise method, in which the reweighted piecewise bound in (34) can itself be minimized as an approximation to $A(\theta)$ (see Sect. 5.1.4). Concretely, suppose that we are using the factor-as-piece approximation. Then we would have a real-valued weight μ_a for each $a \in \mathcal{F}$, and the reweighted approximation is

$$A(\theta; \mathbf{x}) \leq \sum_{a \in \mathcal{F}} \mu_a \log \sum_{\mathbf{y}_a} \exp\left\{ \left(\frac{\theta_a}{\mu_a} \right)^{\top} . f_a(\mathbf{y}_a, \mathbf{x}_a) \right\}. \tag{35}$$

Notice that not only do we take a weighted average of the A_R for each piece, but the weights of each piece are scaled as well. One implication of this is that even uniform μ_a results in a qualitatively different approximation than unweighted piecewise training.

4 Piecewise pseudolikelihood

Although piecewise training breaks apart intractable structures, it can still be computationally expensive when the variables have large cardinality. This is because computing the local normalization functions A_R in (8) requires summing over all assignments to a piece. For example, if all factors are binary, then this summation requires quadratic time in the variable cardinality. Although this is feasible for many models, if the cardinality is large, this cost can be unacceptable.

Pseudolikelihood (Sect. 2), on the other hand, scales linearly in the variable cardinality, because each term in the pseudolikelihood conditions all but one variable in the model. However, on the data that we consider here, PL has considerably worse accuracy than piecewise training. This suggests a hybrid method, in which we apply pseudolikelihoodization to the node-split graph (Sect. 3.2) rather than the original graphical model. We call this training method piecewise pseudolikelihood (PWPL). Surprisingly, we find that two approximations work better than one: on the data considered here, PWPL—which applies the node-split approximation before pseudolikelihood—works better than the pseudolikelihood approximation alone.

This section proceeds as follows. In Sect. 4.1, we describe PWPL in terms of the node-split graph, which was presented previously in Sect. 3.2. This viewpoint allows us to show that under certain conditions, PWPL converges to the piecewise solution in the asymptotic limit of infinite data (Sect. 4.2). In addition, it provides some insight into when PWPL may be expected to do well and to do poorly, an insight that we verify on synthetic and real-world data in Sect. 5.2.

4.1 Definition

In this section, we define piecewise pseudolikelihood. The main motivation of piecewise training is computational efficiency, but in fact PW does not always provide a large gain in training time over other approximate methods. In particular, the time required to evaluate the piecewise likelihood at one parameter setting is the same as is required to run one iteration of belief propagation (BP). More precisely, piecewise training uses $O(m^K)$ time, where m is the maximum number of assignments to a single variable y_s and K is the size of the largest factor. Belief propagation also uses $O(m^K)$ time per iteration; thus, the only computational

savings over BP is a factor of the number of BP iterations required. In tree-structured graphs, piecewise training is no more efficient than forward-backward.

To address this problem, we propose piecewise pseudolikelihood. Piecewise pseudolikelihood (PWPL) is defined as:

$$\ell_{\text{PWPL}}(\Theta; \mathbf{x}, \mathbf{y}) = \sum_{a \in \mathcal{F}} \sum_{s \in a} \log p_{\text{LCL}}(y_s | \mathbf{y}_{a \setminus s}, \mathbf{x}, \theta_a), \tag{36}$$

where p_{LCL} is a locally-normalized score similar to a conditional probability and defined below.

So the piecewise pseudolikelihood is a sum of local conditional log-probabilities. Each variable s participates as the domain of a conditional once for each factor that it neighbors. As in piecewise training, the local conditional probabilities p_{LCL} are not the true probabilities according to the model, but are a quantity computed locally from a single piece (in this case, a single factor). The local probabilities p_{LCL} are defined as

$$p_{\text{LCL}}(y_s | \mathbf{y}_{a \setminus s}, \mathbf{x}, \theta_a) = \frac{\Psi_a(y_s, \mathbf{y}_{a \setminus s}, \mathbf{x}_a)}{\sum_{y'_s} \Psi_a(y'_s, \mathbf{y}_{a \setminus s}, \mathbf{x}_a)}. \tag{37}$$

Then given a data set $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$, we select the parameter setting that maximizes

$$O_{\text{PWPL}}(\theta; D) = \sum_i \ell_{\text{PWPL}}(\theta; \mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \sum_a \frac{\|\theta_a\|^2}{2\sigma^2}, \tag{38}$$

where the second term is a Gaussian prior on the parameters to reduce overfitting. The piecewise pseudolikelihood is convex as a function of θ, and so its maximum can be found using numerical optimization.

For simplicity, we have presented PWPL for the case in which each piece contains exactly one factor. If larger pieces are desired, then simply take the summation over a in (36) to be over pieces rather than over factors, and generalize the definition of p_{LCL} appropriately.

Compared to standard PW, the main advantage of PWPL is that training requires only $O(m)$ time rather than $O(m^K)$. Compared to pseudolikelihood, the difference is that whereas in pseudolikelihood each local term conditions on the entire Markov blanket, in PWPL each local term conditions only on a variable's neighbors within a single factor. For this reason, the local terms in PWPL are not true conditional distributions according to the model. The difference between PWPL and pseudolikelihood is illustrated in Fig. 2. In the next section, we discuss why in some situations this can cause PWPL to have better accuracy than pseudolikelihood.

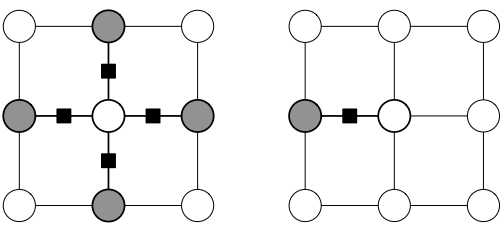

Fig. 2 Illustration of the difference between piecewise pseudolikelihood (PWPL) and standard pseudolikelihood. In standard PL, *at left*, the local term for a variable y_s is conditioned on its entire Markov blanket. In PWPL, *at right*, each local term conditions only on the neighbors within a single factor

4.2 Analysis

PWPL can be readily understood from the node-split viewpoint. In particular, the piecewise pseudolikelihood is simply the standard pseudolikelihood applied to the node-split graph. In this section, we use the asymptotic consistency of standard pseudolikelihood to gain insight into the performance of PWPL.

Suppose that we are given an infinite data set drawn from the distribution p_{NS}, as defined in Sect. 3.2. (In that section, we argued that this yields an equivalent piecewise estimate to the true distribution p^*.) Now, the standard consistency result for pseudolikelihood is that if the model class contains the generating distribution, then the pseudolikelihood estimate converges asymptotically to the true distribution (for more information, see Hyvarinen 2006, Gidas 1988). In this setting, that implies the following statement. If the model family defined by the node-split graph G' contains p_{NS}, then piecewise pseudolikelihood converges in the limit to the same parameter setting as PW. (On the other hand, if there is model mismatch, that is, if the mode class does not contain the generating distribution, then PL does not necessarily converge to a ML solution.)

This asymptotic statement provides no guarantee about how PWPL will perform on finite data. Even so, it has several interesting consequences that provide insight into the method. First, it may impact what sort of model is conducive to PWPL. For example, consider a Potts model with unary factors $\Psi(y_s) = [1 \; e^{\theta_s}]^\top$ for each variable s, and pairwise factors

$$\Psi(y_s, y_t) = \begin{pmatrix} e^{\theta_{st}} & 1 \\ 1 & 1 \end{pmatrix}, \tag{39}$$

for each edge (s, t), so that the model parameters are $\{\theta_s\} \cup \{\theta_{st}\}$. Then the above condition for PWPL to converge in the infinite data limit will never be satisfied, because the pairwise piece cannot represent the marginal distribution of its variables. In this case, PWPL may be a bad choice, or it may be useful to consider pieces that contain more than one factor.

Second, this analysis provides intuition about the differences between piecewise pseudolikelihood and standard pseudolikelihood. For each variable s with neighborhood $N(s)$, standard pseudolikelihood approximates the model marginal $p(y_{N(s)})$ over the neighborhood by the empirical marginal $\tilde{p}(y_{N(s)})$. We expect this approximation to work well when the model is a good fit, and the data is ample. In PWPL, we perform the node-splitting transformation on the graph prior to maximizing the pseudolikelihood. The effect of this is to reduce each variable's neighborhood size, that is, the cardinality of $N(s)$.

This has two potential advantages. First, because the neighborhood size is small, the maximum PWPL estimate may converge to maximum PW estimate faster than pseudolikelihood converges to maximum likelihood. One may reasonably expect maximum likelihood to be better than PW, so whether to prefer standard PL or PWPL depends on precisely how much faster the convergence is. Second, the node-split model may be able to exactly model the marginal of its neighborhood in cases where the original graph may not be able to model its larger neighborhood. Because the neighborhood is smaller, the pseudolikelihood convergence condition may hold in the node-split model when it does not in the original model. In other words, standard pseudolikelihood requires that the original model is a good fit to the full distribution. In contrast, we expect piecewise pseudolikelihood to be a good approximation to PW when each individual piece fits the empirical distribution well. This explains how the double approximation of PWPL could result in a better estimate than PL alone: even though given infinite data we expect PL to be a better estimate than PWPL, PWPL might require less data to reach its limit.

Finally, this analysis suggests that we might expect piecewise pseudolikelihood to perform poorly in two regimes: First, if so much data is available that pseudolikelihood has asymptotically converged, then it makes sense to use pseudolikelihood rather than piecewise pseudolikelihood. Second, if features of the local factors cannot fit the training data well, then we expect the node-split model to fit the data poorly even if the original model was good. In this case, neither piecewise training nor piecewise pseudolikelihood can be expected to do well.

5 Results

In this section, we evaluate piecewise training and its generalizations on several benchmark data sets from natural-language processing. We consider four real-world NLP tasks: part-of-speech tagging, named-entity recognition, noun-phrase chunking, and information extraction.

For *part-of-speech tagging (POS)*, we report results on the WSJ Penn Treebank data set. Results are averaged over five different random subsets of 1911 sentences, sampled from Sections 0–18 of the Treebank. Results are reported from the standard development set of Sections 19–21 of the Treebank. We use a first-order linear chain CRF. There are 45 part-of-speech labels.

In *named-entity recognition*, the task is to find proper nouns in text. We use the CoNLL 2003 data set, consisting of 14,987 newswire sentences annotated with names of people, organizations, locations, and miscellaneous entities. We test on the standard development set of 3,466 sentences. Evaluation is done using precision and recall on the extracted chunks, and we report $F_1 = 2PR/(P+R)$. We use a linear-chain CRF, whose features are described in Table 1.

For the task of *noun-phrase chunking (chunking)*, we use a loopy model, the *factorial CRF* introduced in Sutton et al. (2007). As in that work, we consider the task of jointly predicting part-of-speech tags and segmenting noun phrases on the CoNLL 2000 data set. The structure of this model is a pair of coupled chains. If $\mathbf{w} = (w_1, w_2, \ldots, w_T)$ denotes the part-of-speech (POS) labels and $\mathbf{y} = (y_1, y_2, \ldots, y_T)$ denotes the noun-phrase (NP) labels,

Table 1 Input features $q_k(\mathbf{x}, t)$ for the CoNLL named-entity data. In the above w_t is the word at position t. All features are binary valued

$w_t = v$ for all words v in vocabulary
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t contains a dash
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
The character sequence $c_0 \ldots c_n$ is a prefix of w_t (where $n \in [0, 4]$)
The character sequence $c_0 \ldots c_n$ is a suffix of w_t (where $n \in [0, 4]$)
The character sequence $c_0 \ldots c_n$ occurs in w_t (where $n \in [0, 4]$)
w_t appears in list of first names, last names, countries, locations, honorifics, etc.
$q_k(\mathbf{x}, t+\delta)$ for all k and $\delta \in [-2, 2]$

Table 2 Input features $q_k(\mathbf{x}, t)$ for the noun-phrase segmentation data. In the above w_t is the word at position t. All features are binary valued

$w_t = v$ for all words v in vocabulary
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
w_t matches `.*[0-9].*`
w_t appears in list of first names, last names, company names, days, months, or geographic entities
w_t is contained in a lexicon of words with POS T (from Brill tagger)
$T_t = T$ for all parts of speech T
$q_k(\mathbf{x}, t+\delta)$ for all features q_k listed above and $\delta \in [-3, 3]$

then the model is

$$p(\mathbf{y}, \mathbf{w}|\mathbf{x}) = \prod_{t=1}^{T} \Psi_0(y_t, y_{t-1}, \mathbf{x}) \Psi_1(w_t, w_{t-1}, \mathbf{x}) \Psi_2(w_t, y_t, \mathbf{x}), \tag{40}$$

where each type of factor Ψ_0, Ψ_1, Ψ_2 is expressed in the log-linear form of (2). The features used are described in Table 2. There are 45 different POS labels, and the three NP labels. We report F1 on noun-phrase chunks. In previous work, this model was optimized by approximating the partition function using belief propagation, but this was quite expensive. Training on the full data set of 8936 sentences required about 12 days of CPU time.

Finally, for the task of *information extraction*, we consider a model with many irregular loops, which is the skip chain model introduced in Sutton and McCallum (2004). This model incorporates certain long-distance dependencies between word labels into a linear-chain model for information extraction. The idea is that when the same word appears multiple times in the same message, it tends to have the same label. We represent this by adding edges between output nodes (y_i, y_j) when the words at positions i and j are identical and capitalized.

For a sentence $\mathbf{x}$, let $\mathcal{I} = \{(u, v)\}$ be the set of all pairs of sequence positions for which there are skip edges. For example, in the experiments reported here, $\mathcal{I}$ is the set of indices of all pairs of identical capitalized words. Then the conditional probability is modeled as

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, \mathbf{x}) \prod_{(u,v)\in\mathcal{I}} \Psi_{uv}(y_u, y_v, \mathbf{x}), \tag{41}$$

where Ψ_t are the factors for linear-chain edges, and Ψ_{uv} are the factors over skip edges. Each of these factors takes the exponential form in (2).

We use a standard data set of seminar announcements (Freitag 1998). Consistently with the previous work on this data set, we use 10-fold cross validation with a 50/50 training/test split. We report per-token F1 on the speaker and location fields, the most difficult of the four fields. The features used are described in Table 3. Most documents contain many crossing skip-edges, so that exact maximum-likelihood training using junction tree is completely infeasible, so instead we compare to approximate training using loopy belief propagation.

Table 3 Input features $q_k(\mathbf{x}, t)$ for the seminars data. In the above w_t is the word at position t. The "appears to be" features are based on hand-designed regular expressions that can span several tokens. All features are binary valued

$w_t = v$ for all words v in vocabulary
w_t matches `[A-Z][a-z]+`
w_t matches `[A-Z][A-Z]+`
w_t matches `[A-Z]`
w_t matches `[A-Z]+`
w_t matches `[A-Z]+[a-z]+[A-Z]+[a-z]`
w_t appears in list of first names, last names, honorifics, etc.
w_t appears to be part of a time followed by a dash
w_t appears to be part of a time preceded by a dash
w_t appears to be part of a date
$q_k(\mathbf{x}, t+\delta)$ for all features q_k listed above and $\delta \in [-4, 4]$

For all the data sets, we compare to pseudolikelihood and conditional maximum likelihood with belief propagation. All of these objective functions are maximized using limited-memory BFGS. We use a Gaussian prior with variance $\sigma^2 = 10$. In previous work, the prior parameter had been tuned on each data set for belief propagation, and for the local models we used the same prior parameter without change. At test time, prediction is always performed using max-product belief propagation.

5.1 Piecewise training

In this section, we compare piecewise training to both maximum likelihood and pseudolikelihood. To be as fair as possible to pseudolikelihood, we compare to two variations of pseudolikelihood, one based on nodes and a structured version based on edges. As normally applied, pseudolikelihood is a product of per-node conditional probabilities, as in (5). But this per-variable pseudolikelihood function does not work well for sequence labeling, because of the strong interactions between neighboring sequence positions. In order to have a stronger baseline, we also compare to a pairwise version of pseudolikelihood:

$$\ell_{\text{EPL}}(\theta; \mathbf{x}, \mathbf{y}) = \log \prod_{st} p(y_s, y_t | \mathbf{y}_{N(s,t)}), \tag{42}$$

where the variables s, t range over all variables that share a factor. That is, instead of using local conditional distributions over single variables, we use distributions over pairs of variables, hoping to take more of the sequential interactions into account.

5.1.1 Named-entity recognition (linear-chain CRF)

First, we evaluate the accuracy of piecewise training on a tractable model, so that we can compare the accuracy to exact maximum-likelihood training. We use a linear-chain CRF for the named-entity recognition task. The results are shown in Table 4. Piecewise training performs better than either of the pseudolikelihood methods. Furthermore, even though it is a completely local training method, piecewise training performs comparably to exact CRF training.

Now, in a linear-chain model, piecewise training has the same computational complexity as exact CRF training, so we do not mean to advocate the piecewise approximation for linear-chain graphs. Rather, that the piecewise approximation loses no accuracy on the linear-chain model is encouraging when we turn to loopy models.

Table 4 Comparison of piecewise training to exact and pseudolikehood training on a linear-chain CRF for named-entity recognition. On this tractable model, piecewise methods are more accurate than pseudolikelihood, and just as accurate as exact training

Method	Overall F1
PW	**91.2**
Pseudolikelihood	84.7
Per-edge PL	89.7
Exact	90.6

Table 5 Comparison of piecewise training to other methods on a two-level factorial CRF for joint part-of-speech tagging and noun-phrase segmentation. Shown are the means and standard deviations over five random subsets of the training data

Method	Noun-phrase F1
Piecewise	**88.1** ±0.48
Pseudolikelihood	84.9 ±0.39
Per-edge PL	86.5 ±0.64
BP	86.0 ±0.39

Table 6 Comparison of piecewise training to other methods on a skip-chain CRF for seminar announcements. Shown are the means and standard deviations over 10-fold cross-validation

Method	Token F1 location	Speaker
Piecewise	**87.7** ±1.2	75.4 ±2.5
Pseudolikelihood	67.1 ±4.8	25.5 ±4.2
Per-edge PL	76.9 ±4.1	69.3 ±3.0
BP	86.6 ±1.5	**78.2** ±1.8

5.1.2 Noun-phrase chunking (factorial CRF)

In this section, we compare piecewise training to pseudolikelihood and maximum likelihood on the factorial CRF for the noun-phrase chunking task. We report results here on subsets of 223 training sentences, and the standard test set of 2012 sentences. Results are averaged over 5 different random subsets. Results on this loopy data set are presented in Table 5. Again, the piecewise estimator performs better both than either version of pseudolikelihood and than maximum-likelihood estimation using belief propagation. On this small subset, approximate maximum likelihood training with BP requires 1.8 h, but piecewise training is still twice as fast, using 0.83 h.

5.1.3 Information extraction (skip-chain CRF)

Results on this model are given in Table 6. Pseudolikelihood performs particularly poorly on this model. Piecewise estimation performs much better, but worse than approximate training using BP.

Piecewise training is faster than loopy BP: in our implementation piecewise training used on average 3.5 hr, while loopy BP used 6.8 hr. To get these loopy BP results, however, we must carefully initialize the training procedure from the linear-chain parameters, as

Table 7 Comparison of basic piecewise training to reweighted piecewise bound with uniform μ

Model	Basic	Reweighted
Linear-chain	91.2	90.4
FCRF	88.1	86.4
Skip-chain (location)	87.7	75.5
Skip-chain (speaker)	75.4	69.2

discussed in Sutton (2008). For example, if instead we initialize the model to the uniform distribution, not only does loopy BP training take much longer, over 10 hours, but testing performance is much worse, because the convex optimization procedure has difficulty with noisier gradients. With uniform initialization, loopy BP does not converge for all training instances, especially at early iterations of training. Carefully initializing the model parameters seems to alleviate these issues, but this model-specific tweaking was unnecessary for piecewise training.

5.1.4 Reweighted piecewise training

In this section, we evaluate a simple version of reweighted piecewise training (Sect. 3.5; (35)). in which the pieces are weighted by a convex combination. The performance of reweighted piecewise training with uniform μ_a is presented in Table 7. In all cases, the reweighted piecewise method performs worse than the basic piecewise method.

What seems be happening is that in each of these models, there are several hundred edges, so that the weight μ_a for each region is rather small, e.g., about 0.01. Because the reweighting happens in the log domain (see (35)), the small weights have a strong annealing effect, so that the summation over $\mathbf{y}_a$ in (35) is dominated by the assignment that has the maximum value of $\theta_a^\top f_a(\mathbf{y}_a, \mathbf{x}_a)$.

To be clear, these results are for a single, fixed value of the per-piece weights $\{\mu_a\}$. Optimizing the bound with respect to all the $\{\mu_a\}$ would presumably perform better, although at a larger computational cost. It is also possible that more general reweighting schemes, that do not require the weights to sum to one, may perform better. For some results in this vein, see Shotton et al. (2006).

5.2 Piecewise pseudolikelihood

In this section, we compare PWPL to both regular piecewise and pseudolikelihood on both synthetic and real-world data. On synthetic data (Sect. 5.2.1), we find that PWPL works better than PL for small amounts of training data, but PL works better when the training set is larger, confirming the intuition resulting from the asymptotic arguments of Section 4.2. Second, on real-world NLP data sets (Sect. 5.2.2), we find that PWPL performs often comparably to piecewise training and to maximum likelihood, and always better than pseudolikelihood. Furthermore, PWPL can be as much as ten times faster than batch CRF training.

5.2.1 Synthetic data

In Sect. 4.2, we argued intuitively that PWPL may perform better on small data sets, and pseudolikelihood on larger ones. In this section we verify this intuition in experiments on synthetic data. The general setup is replicated from Lafferty et al. (2001). We generate data from a second-order HMM with transition probabilities

$$p_\alpha(y_t|y_{t-1}, y_{t-2}) = \alpha p_2(y_t|y_{t-1}, y_{t-2}) + (1-\alpha)p_1(y_t|y_{t-1}) \tag{43}$$

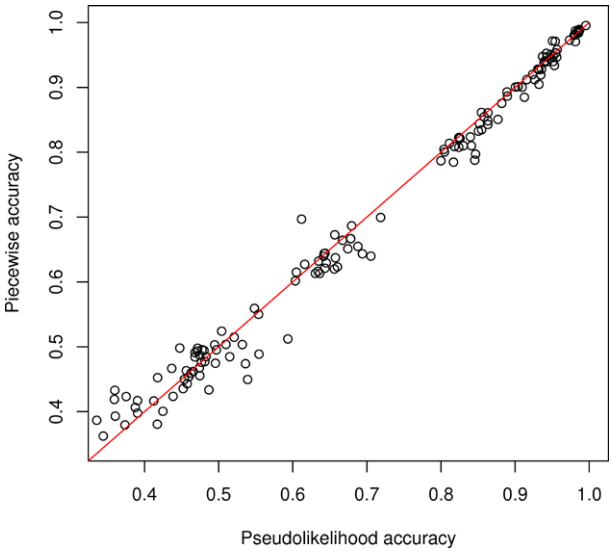

Fig. 3 Comparison of piecewise to pseudolikelihood on synthetic data. Pseudolikelihood has slightly better accuracy on training instances than piecewise (PW and PWPL perform exactly the same; this is not shown)

and emission probabilities

$$p_\alpha(x_t|y_t, x_{t-1}) = \alpha p_2(x_t|y_t, x_{t-1}) + (1-\alpha)p_1(x_t|y_t). \tag{44}$$

Thus, for $\alpha = 0$, the generating distribution p_α is a first-order HMM, and for $\alpha = 1$, it is an autoregressive second-order HMM. We compare different approximate methods for training a first-order CRF. Therefore higher values of α make the learning problem more difficult, because the model family does not contain second-order dependencies. We use five states and 26 possible observation values. For each setting of α, we sample 25 different generating distributions. From each generating distribution we sample 1,000 training instances of length 25, and 1,000 testing instances. We use $\alpha \in \{0, 0.1, 0.25, 0.5, 0.75, 1.0\}$, for 150 synthetic generating models in all.

First, we find that piecewise pseudolikelihood performs almost identically to PW. Averaged over the 150 data sets, the mean difference in testing error between piecewise pseudolikelihood and piecewise is 0.002, and the correlation is 0.999.

Second, we compare PWPL to traditional pseudolikelihood. On this data, pseudolikelihood performs slightly better overall, but the difference is not statistically significant (paired t-test; $p > 0.1$). However, when we examine the accuracy as a function of training set size (Fig. 4), we notice an interesting two-regime behavior. Both PWPL and pseudolikelihood seem to be converging to a limit, and the eventual pseudolikelihood limit is higher than PWPL, but PWPL converges to its limit faster. This is exactly the behavior intuitively predicted by the argument in Sect. 4.2: that PWPL can converge to the piecewise solution in less training data than pseudolikelihood to its (potentially better) solution.

Of course, the training set sizes considered in Fig. 4 are fairly small, but this is exactly the case we are interested in, because on natural language tasks, even when hundreds of thousands of words of labeled data are available, this is still a small amount of data compared to the number of useful features.

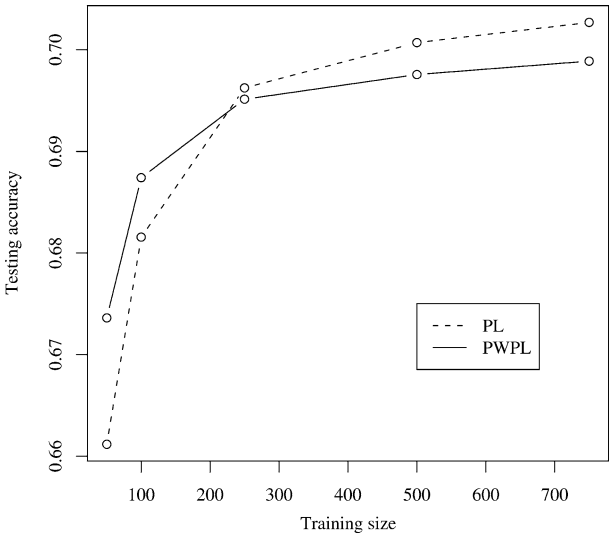

Fig. 4 Learning curves for PWPL and pseudolikelihood. For smaller amounts of training data PWPL performs better than pseudolikelihood, but for larger data sets, the situation is reversed

5.2.2 Real-world data

Now we compare piecewise training and piecewise pseudolikelihood on the NLP tasks described earlier: POS tagging, NER, chunking, and information extraction. For the POS tagging task, we use five randomly-chosen subsets of 1911 sentences for training. For the NER task, we use the standard training and test split in the CoNLL 2003 data set; we report results on the development set. For the chunking task, we use five randomly-chosen subsets of 447 sentences for training, and the standard test set. Finally, for the information extraction task, we use 10-fold cross validation with a 50/50 training-test split.

On the first three tasks—part-of-speech tagging, chunking, and NER—piecewise pseudolikelihood and standard piecewise training have equivalent accuracy both to each other and to maximum likelihood (Table 8). Despite this, piecewise pseudolikelihood is much more efficient than PW (Table 8). On the named-entity data, which has the fewest labels, PWPL uses 75% of the time of PW, a modest improvement. On the data sets with more labels, the difference is more dramatic: on the POS data, PWPL uses 16% of the time of piecewise and on the chunking data, PWPL needs only 13%. Similarly, PWPL is also between is 5 to 10 times faster than maximum likelihood.

The training times of the baseline methods may appear relatively modest. If so, this is because for both the chunking and POS data sets, we use relatively small subsets of the full training data, to make running this comparison more convenient. This makes the absolute difference in training time even more meaningful than it may appear at first. Also, it may appear from Table 8 that PWPL is faster than standard pseudolikelihood, but the apparent difference is due to low-level inefficiencies in our implementation. In fact the two algorithms have similar complexity.

On the skip chain data (Table 9), PW performs worse than exact training using BP, and piecewise pseudolikelihood performs worse than PW. Both piecewise methods, however, perform better than pseudolikelihood.

As predicted in Sect. 4.2, pseudolikelihood is indeed a better approximation on the node-split graph. In Table 8, PL performs much worse than maximum likelihood (ML), but PWPL performs only slightly worse than PW. In Table 9, the difference between PWPL and PW is larger, but still less than the difference between PL and ML.

Table 8 Comparison of piecewise pseudolikelihood to PW and to pseudolikelihood on real-world NLP tasks. Piecewise pseudolikelihood is in all cases comparable to piecewise, and on two of the data sets superior to pseudolikelihood

	ML	PL	PW	PWPL
POS				
Accuracy	94.4 ±0.07	94.4 ±0.05	94.3 ±0.08	94.4 ±0.07
Time (s)	33846 ±6800	6705 ±740	23537 ±8700	**3911** ±1500
Chunking				
Accuracy	91.4 ±0.21	90.3 ±0.19	91.7 ±0.14	91.4 ±0.12
Time (s)	24769 ±6183	1534 ±165	5708 ±701	**766** ±83
Named-entity				
Chunk F1	90.5	85.1	90.5	90.3
Time (s)	52396	8651	6311	**4780**

Table 9 F1 performance of PWPL, piecewise, and pseudolikelihood on information extraction from seminar announcements. Both the mean and standard deviation over the 10 folds are shown. Both PW and piecewise pseudolikelihood outperform pseudolikelihood

	BP	PL	PW	PWPL
START-TIME	96.5 ±2.3	82.2 ±6.1	97.1 ±0.42	94.1 ±0.78
END-TIME	95.9 ±2.4	73.4 ±3.8	96.5 ±0.49	90.4 ±2.7
LOCATION	85.8 ±5.2	73.0 ±2.1	88.1 ±1.2	85.3 ±1.6
SPEAKER	74.5 ±5.7	27.9 ±3.1	72.7 ±3.1	65.0 ±5.2

6 Related work

Because the piecewise estimator is such an intuitively appealing method, it has been used in several scattered places in the literature, for tasks such as information extraction (Wellner et al. 2004), collective classification (Greiner et al. 2005), and computer vision (Freeman et al. 2000). In these papers, the piecewise method is reported as a successful heuristic for training large models, but its performance is not compared against other training methods. We are unaware of previous work systematically studying this procedure in its own right.

Local training methods have recently been the subject of much interest (Abbeel et al. 2005; Toutanova et al. 2003; Punyakanok et al. 2005). The local training method most closely connected to the current work is pseudolikelihood (Besag 1975, 1977). The difference in the current work is that regular piecewise training does not condition on neighboring nodes, but ignores them altogether during training. This is depicted schematically by the factor graphs in Fig. 5. In pseudolikelihood, each locally-normalized term for a variable or edge in pseudolikelihood includes contributions from a number of factors that connect to the neighbors whose observed values are taken from labeled training data. All these factors are circled in the top section of Fig. 5. In piecewise training, each factor becomes an independently, locally-normalized term in the objective function. PWPL, on the other hand, conditions on some of the variables in the Markov blanket, but not on all of them, like standard pseudolikelihood. We are unaware of previous variants of pseudolikelihood that condition on less than the full Markov blanket.

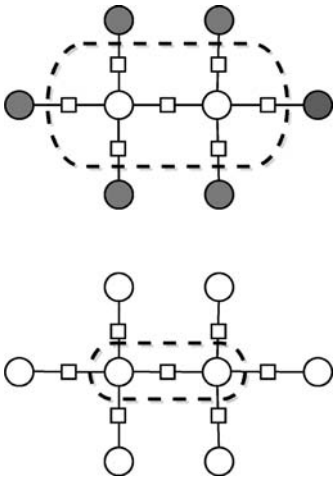

Fig. 5 Schematic factor-graph depiction of the difference between pseudolikelihood (*top*) and piecewise training (*bottom*). Each term in pseudolikelihood normalizes the product of many factors (as *circled*), while piecewise training normalizes over one factor at a time

Huang and Ogata (1999) consider the asymptotic properties of generalized pseudolikelihood estimators whose local conditional terms predict subgraphs of the model rather than single nodes. A special case of this is the pairwise pseudolikelihood that we compare against in Sect. 5.1.

Also, in statistics there has been work on general families of surrogate likelihoods, called composite likelihoods, which are sums of marginal or conditional log likelihoods (Lindsay 1988; Cox and Reid 2004). Such composite likelihoods are consistent and asymptotically normal under relatively general assumptions. An example of using a composite likelihood on structured models for natural-language data is given by Kakade et al. (2002). But these are designed for a different situation than ours, namely when the joint likelihoods are difficult to compute but marginal likelihoods are easier to work with. An example of this situation is the multivariate Gaussian. In our context, marginal likelihoods are difficult to compute, so composite likelihoods are not as useful. The piecewise likelihood is not a type of composite likelihood, because in the likelihood of each piece, the contribution of the rest of the model is ignored, not marginalized out.

Independently, Choi et al. (2007) present a node-splitting technique for upper bounds during inference, which is closely related to the piecewise upper bound that we use here for learning.

An interesting connection exists between piecewise pseudolikelihood and maximum entropy Markov models (MEMMs) (Ratnaparkhi 1996; McCallum et al. 2000). In a linear chain with variables $y_1 \dots y_T$, we can write the piecewise pseudolikelihood as

$$\ell_{\text{PWPL}}(\theta; \mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \log p_{\text{LCL}}(y_t | y_{t-1}, \mathbf{x}) p_{\text{LCL}}(y_{t-1} | y_t, \mathbf{x}). \tag{45}$$

The first part of (45) is exactly the likelihood for an MEMM, and the second part is the likelihood of a backward MEMM. Although this viewpoint is interesting, it does not really explain the performance of PWPL, for two reasons. First, on NLP tasks, backward MEMMs tend to perform worse than forward MEMMs, so it is not immediately clear why combining the likelihoods of a forward and a backward MEMM would work better than either one individually. The connection to pseudolikelihood and piecewise training seems to be more illuminating in this regard. Second, the local normalization in MEMMs is part of the model,

not part of the training procedure. To explain what we mean by this, consider a forward-only MEMM, that is,

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} p_{\mathrm{LCL}}(y_t|y_{t-1}, \mathbf{x}) = \prod_{t=1}^{T} \frac{\Psi_t(y_t, y_{t-1}, \mathbf{x})}{\sum_{y'_t} \Psi_t(y'_t, y_{t-1}, \mathbf{x})}. \tag{46}$$

Each factor $p_{\mathrm{LCL}}(y_t|y_{t-1}, \mathbf{x})$ contains a local normalizer, which sums only on y_t (these are the denominators in (46)). These local normalization factors are considered part of the model; concretely, this means that when we receive a new input $\mathbf{x}'$ from the test data, the maximum-probability assignment is

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \prod_{t=1}^{T} \frac{\Psi_t(y_t, y_{t-1}, \mathbf{x})}{\sum_{y'_t} \Psi_t(y'_t, y_{t-1}, \mathbf{x})}. \tag{47}$$

In contrast, in PWPL, the local normalization factors are not considered part of the model. For example, to make an analog to (45), in a linear-chain CRF trained by PWPL, the maximum-probability assignment for a test input $\mathbf{x}$ is still computed as

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, \mathbf{x}). \tag{48}$$

That is, the local normalization factors are removed at test time. MEMMs crucially depend on including the local normalization factors at test time: In an MEMM, if the local normalization factors are dropped at test time, then the predictions are very poor. Without the connections to piecewise training and to pseudolikelihood, it is not clear why normalization at test time is required when training a forward MEMM but not when training using (45).

Piecewise pseudolikelihood also has an interesting connection to search-based learning methods (Daumé and Marcu 2005). Such methods learn a model to predict the next state of a local search procedure from a current state. Typically, training is viewed as classification, where the correct next states are positive examples, and alternative next states are negative examples. One view of the current work is that it incorporates backward training examples, that attempt to predict the *previous* search state given the current state.

Finally, stochastic gradient methods, which make gradient steps based on subsets of the data, have recently been shown to converge significantly faster for CRF training than batch methods, which evaluate the gradient of the entire data set before updating the parameters (Vishwanathan et al. 2006). Stochastic gradient methods are currently the method of choice for training linear-chain CRFs, especially when the data set is large and redundant. However, stochastic gradient methods can also be used to optimize both standard piecewise likelihood and piecewise pseudolikelihood. Thus, although the training time of our baselines could likely be improved considerably, the same is true of our new approaches, so that our comparison is fair.

Also, in some cases, such as in relational learning problems, the data are not iid, and the model includes explicit dependencies between the training instances. For such a model, it is unclear how to apply stochastic gradient, but piecewise pseudolikelihood may still be useful. Finally, stochastic gradient methods do not address cases in which the variables have large cardinality, or when the graphical structure of a single training instance is intractable.

One challenge in the piecewise method is that it is not immediately clear how to extend it to the case of latent variables, because if the same latent variable occurs in different pieces, it

may have different semantics. Recently, Liang et al. (2006, 2008) have presented a method in a similar spirit to piecewise training, but that handles latent variables. They add an additional term to the objective function that encourages pieces that share a latent variable to agree on its distribution. They present two different EM-style algorithms for optimizing this objective function efficiently.

Both likelihood-based methods and max-margin methods require performing inference during training, so it is natural to wonder whether the methods presented in this paper can be adapted to loopy max-margin models. A suggestive step in this direction is *factorized MIRA* (McDonald et al. 2005), in which the margin constraints are required to hold only over single edges, rather than the entire prediction. On a dependency parsing task, this method had good accuracy, but it did not improve training time because the model had special structure that made it amenable to exact inference. It may be interesting to see whether analogs of piecewise methods work well for max-margin training on loopy models.

Earlier versions of the current work have appeared in two conference papers (Sutton and McCallum 2005, 2007b).

Finally, an important future direction is approximate training methods that are mostly local, but that incorporate a limited amount of influence from the rest of the model. An interesting first step in this direction is Ganapathi et al. (2008), which derives a two-loop local training algorithm by considering the maximum-entropy dual of the maximum-likelihood problem. Their method generalizes piecewise training in two ways: first, extra features are added to the model in order to force singe-variable marginals to be consistent with factor marginals; and second, a scalar correction is added to the logarithm of each factor to partially correct for overcounting in the basic piecewise procedure.

7 Conclusion

This paper has presented piecewise training, an intuitively appealing procedure that separately trains factor subsets, called pieces, of a loopy graph. We show that this procedure can be justified as maximizing a loose bound on the log likelihood. On three real-world language tasks with different model structures, piecewise training outperforms several versions of pseudolikelihood, a traditional local training method. On two of the data sets, in fact, piecewise training is more accurate than global training using belief propagation.

Second, we have introduced an extension called piecewise pseudolikelihood, that is especially attractive when the variables in the model have large cardinality. Because PWPL conditions on fewer variables, it can have better accuracy than standard pseudolikelihood, and is dramatically more efficient than the standard piecewise likelihood, requiring as little as 13% of the training time.

Many properties of piecewise training remain to be explored. Our results indicate that in some situations piecewise training should replace pseudolikelihood as the local training method of choice. In particular, the experiments here all used discriminative training, which make local training easier because of the large amount of information in the conditioning variables. In the data sets here, the local features, such as the word identity, provide a large amount of information about the labels in their own right. In generative training, there may be less local information, making piecewise training much less effective. On the other hand, from the exponential family perspective, piecewise training does still match expected statistics of a subgraph to the empirical distribution, which still seems intuitively appealing. For this reason, it is hard to give a definitive characterization of when piecewise training is expected to work well or poorly.

A possible explanation for the performance of piecewise training is that it acts as a form of additional regularization, in that the objective function disfavors parameter settings that obtain good joint likelihood by using long-distance effects of weights. For this reason, connections to generalization bounds for feature selection, some of which take into account the amount of computation, may be interesting.

Finally, a natural question is how readily the methods here extend to the case where pieces are treated as regions that are larger than a single factor. The main challenges here seem to lie in first, how to handle the overlaps among larger pieces, and second, how to choose the pieces. The connection to the Bethe energy may be illuminating here.

Acknowledgements We thank Tom Minka and Martin Szummer for many helpful conversations during an internship by the first author at Microsoft Research Cambridge. We thank an anonymous reviewer, and independently Drew Bagnell and David Blei, for pointing out the simple proof of Proposition 1. (An early draft had a more complicated proof.) Also, we thank David Rosenberg for alerting us an error in an earlier version of this paper. Finally, we thank the members of Sutton's dissertation committee—Tommi Jaakkola, Erik Learned-Miller, Jonathan Machta, and Sridhar Mahadevan—for valuable comments on this work.

This work was supported in part by the Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07-D-0185/0004, and in part by UPenn NSF medium IIS-0803847, and in part by Army prime contract number W911NF-07-1-0216 and University of Pennsylvania subaward number 103-548106. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily react those of the sponsor.

References

Abbeel, P., Koller, D., & Ng, A. Y. (2005). Learning factor graphs in polynomial time and sample complexity. In *Twenty-first conference on uncertainty in artificial intelligence (UAI05)*.

Bernal, A., Crammer, K., Hatzigeorgiou, A., & Pereira, F. (2007). Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Computational Biology*, 3(3).

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, *24*(3), 179–195.

Besag, J. (1977). Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, *64*(3), 616–618.

Choi, A., Chavira, M., & Darwiche, A. (2007). Node splitting: a scheme for generating upper bounds in Bayesian networks. In *Conference on uncertainty in artificial intelligence (UAI)*.

Cox, D. R., & Reid, N. (2004). A note on pseudolikelihood constructed from marginal densities. *Biometrika*, *91*, 729–737.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, *3*, 951–991.

Daumé III, H., & Marcu, D. (2005). Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine Learning (ICML)*, Bonn, Germany.

Finkel, J. R., Manning, C. D., & Ng, A. Y. (2006). Solving the problem of cascading errors: approximate Bayesian inference for linguistic annotation pipelines. In *Conference on empirical methods in natural language processing (EMNLP)*.

Freeman, W. T., Pasztor, E. C., & Carmichael, O. T. (2000). Learning low-level vision. *International Journal of Computer Vision*, *40*(1), 24–57.

Freitag, D. (1998). *Machine learning for information extraction in informal domains*. PhD thesis, Carnegie Mellon University.

Ganapathi, V., Vickrey, D., Duchi, J., & Koller, D. (2008). Constrained approximate maximum entropy learning. In *Conference on uncertainty in artificial intelligence (UAI)*.

Gidas, B. (1988). Consistency of maximum likelihood and pseudolikelihood estimators for Gibbs distributions. In W. Fleming & P.-L. Lions (Eds.), *Stochastic differential systems, stochastic control theory and applications*. New York: Springer.

Greiner, R., Guo, Y., & Schuurmans, D. (2005). Learning coordination classifiers. In *International joint conference on artificial intelligence (IJCAI)*.

Huang, F., & Ogata, Y. (1999). Improvements of the maximum pseudo-likelihood estimators in various spatial statistical models. *Journal of Computational and Graphical Statistics*, *8* (3), 510–530, ISSN 10618600. URL http://www.jstor.org/stable/1390872.

Hyvarinen, A. (2006). Consistency of pseudolikelihood estimation of fully visible Boltzmann machines. *Neural Computation*, *18*(10), 2283–2292.

Kakade, S., Teh, Y. W., & Roweis, S. (2002). An alternative objective function for Markovian fields. In *Proceedings of the nineteenth international conference on machine learning*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *International conference on machine learning (ICML)*.

Li, S. Z. (2001). *Markov random field modeling in image analysis*. New York: Springer.

Liang, P., Taskar, B., & Klein, D. (2006). Alignment by agreement. In *Human language technology and North American association for computational linguistics (HLT/NAACL)*.

Liang, P., Klein, D., & Jordan, M. I. (2008). Agreement-based learning. In *Advances in neural information processing systems (NIPS)*.

Lindsay, B. G. (1988). Composite likelihood methods. *Contemporary Mathematics*, 221–239.

McCallum, A., & Wellner, B. (2005). Conditional models of identity uncertainty with application to noun coreference. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems 17* (pp. 905–912). Cambridge: MIT Press.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. In *International conference on machine learning (ICML)* (pp. 591–598). San Francisco: Morgan Kaufmann.

McDonald, R., Crammer, K., & Pereira, F. (2005). *Spanning tree methods for discriminative training of dependency parsers* (Technical Report MS-CIS-05-11). University of Pennsylvania CIS.

Minka, T. (2001a). Expectation propagation for approximate Bayesian inference. In *17th conference on uncertainty in artificial intelligence (UAI)* (pp. 362–369).

Minka, T. P. (2001b). *The EP energy function and minimization schemes*. http://research.microsoft.com/~minka/papers/ep/minka-ep-energy.pdf.

Minka, T. (2005). *Divergence measures and message passing* (Technical Report MSR-TR-2005-173). Microsoft Research.

Parise, S., & Welling, M. (2005). Learning in Markov random fields: an empirical study. In *Joint Statistical Meeting (JSM2005)*.

Punyakanok, V., Roth, D., Yih, W.-T., & Zimak, D. (2005). Learning and inference over constrained output. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)* (pp. 1124–1129).

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Conference on empirical methods in natural language proceeding (EMNLP)*.

Rosen-Zvi, M., Yuille, A. L., & Jordan, M. I. (2005). The dlr hierarchy of approximate inference. In *Conference on uncertainty in artificial intelligence (UAI)*.

Shotton, J., Winn, J., Rother, C., & Criminisi, A. (2006). Textonboost: joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision (ECCV)*.

Stern, D. H., Graepel, T., & MacKay, D. J. C. (2005). Modelling uncertainty in the game of go. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems 17* (pp. 1353–1360). Cambridge: MIT Press.

Sutton, C. (2008). *Efficient training methods for conditional random fields*. PhD thesis. University of Massachusetts.

Sutton, C., & McCallum, A. (2004). Collective segmentation and labeling of distant entities in information extraction. In *ICML workshop on statistical relational learning and its connections to other fields*.

Sutton, C., & McCallum, A. (2005). Piecewise training of undirected models. In *Conference on uncertainty in artificial intelligence (UAI)*.

Sutton, C., & McCallum, A. (2007a). An introduction to conditional random fields for relational learning. In L. Getoor & B. Taskar (Eds.), *Introduction to statistical relational learning*. Cambridge: MIT Press.

Sutton, C., & McCallum, A. (2007b). Piecewise pseudolikelihood for efficient CRF training. In *International conference on machine learning (ICML)*.

Sutton, C., & Minka, T. (2006). *Local training and belief propagation* (Technical Report TR-2006-121). Microsoft Research.

Sutton, C., Rohanimanesh, K., & McCallum, A. (2004). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *International conference on machine learning (ICML)*.

Sutton, C., McCallum, A., & Rohanimanesh, K. (2007). Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, *8*, 693–723.

Taskar, B., Guestrin, C., & Koller, D. (2004a). Max-margin Markov networks. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge: MIT Press.

Taskar, B., Klein, D., Collins, M., Koller, D., & Manning, C. (2004b). Max-margin parsing. In *Empirical methods in natural language processing (EMNLP04)*.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL*.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., & Murphy, K. (2006). Accelerated training of conditional random fields with stochastic meta-descent. In *International conference on machine learning (ICML)* (pp. 969–976).

Wainwright, M. J., Jaakkola, T., & Willsky, A. S. (2002). A new class of upper bounds on the log partition function. In *Uncertainty in artificial intelligence*.

Wainwright, M. J., Jaakkola, T., & Willsky, A. S. (2003a). Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, *45*(9), 1120–1146.

Wainwright, M. J., Jaakkola, T., & Willsky, A. S. (2003b). Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching. In *Ninth workshop on artificial intelligence and statistics*.

Wellner, B., McCallum, A., Peng, F., & Hay, M. (2004). An integrated, conditional model of information extraction and coreference with application to citation graph construction. In *20th conference on uncertainty in artificial intelligence (UAI)*.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, *51*(7), 2282–2312.